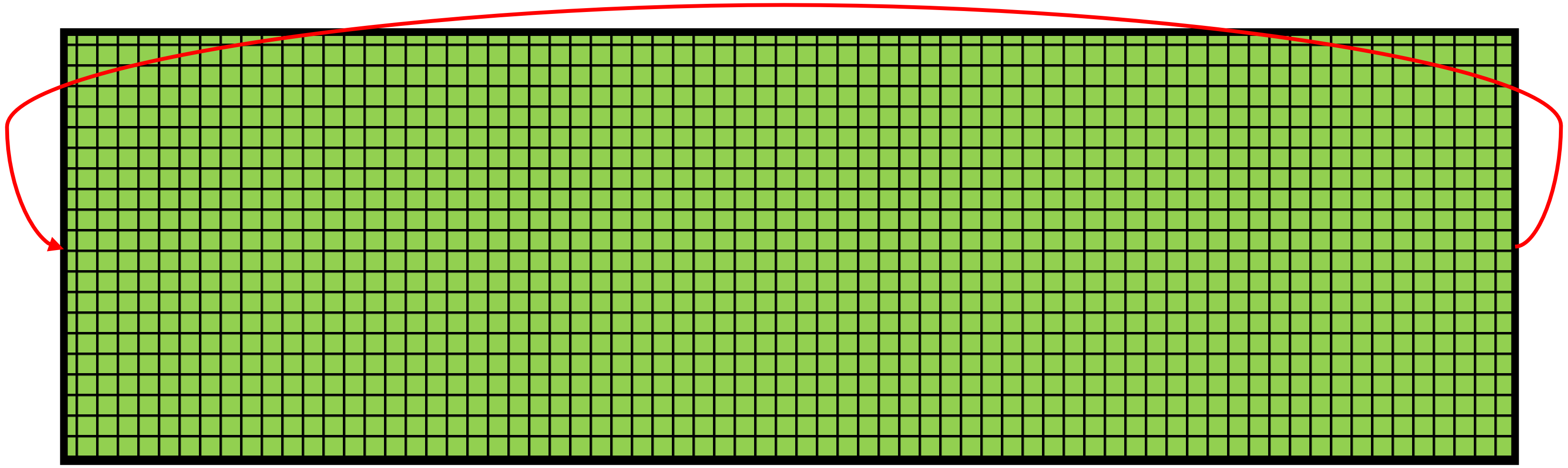
Adventures in CFD and Geometry

3DTea (Finally!!)

21-Nov-2024

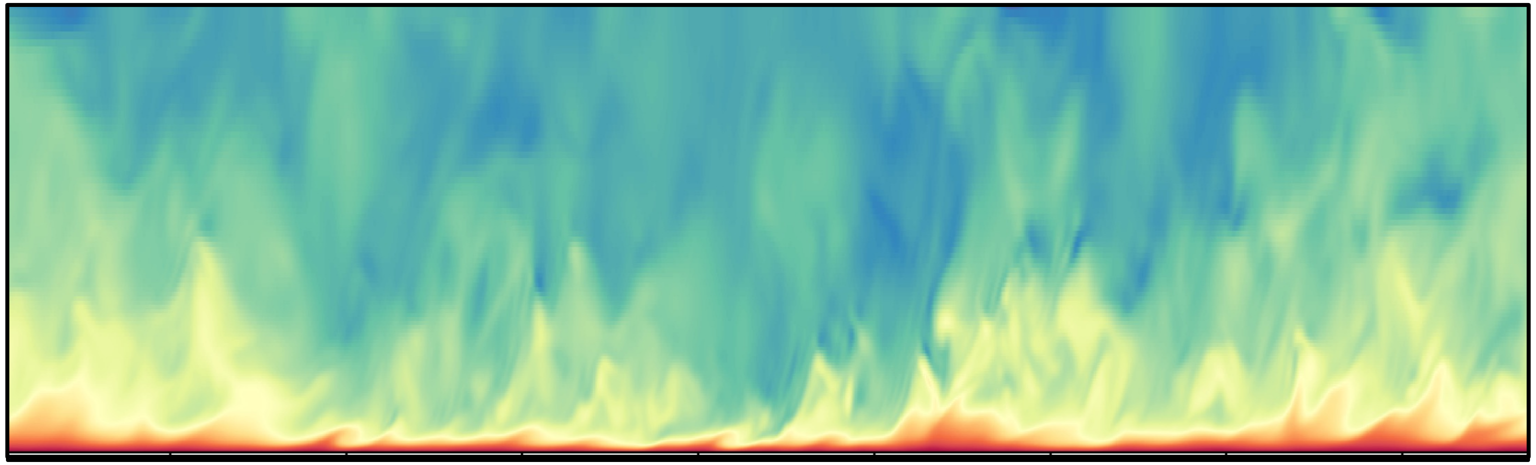
Picture generated using ChatGPT
Prompt – title of the talk

Part 1: Fake It Till You Make It



Standard Flow Configuration

How to best populate the grid cell values, to reach the final state “as quickly as possible”?

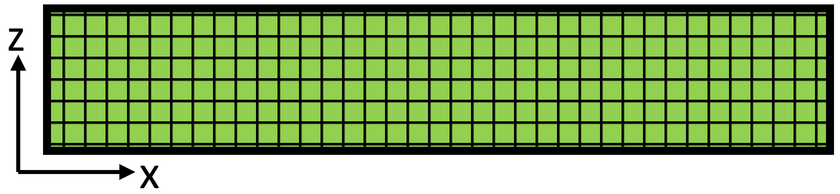


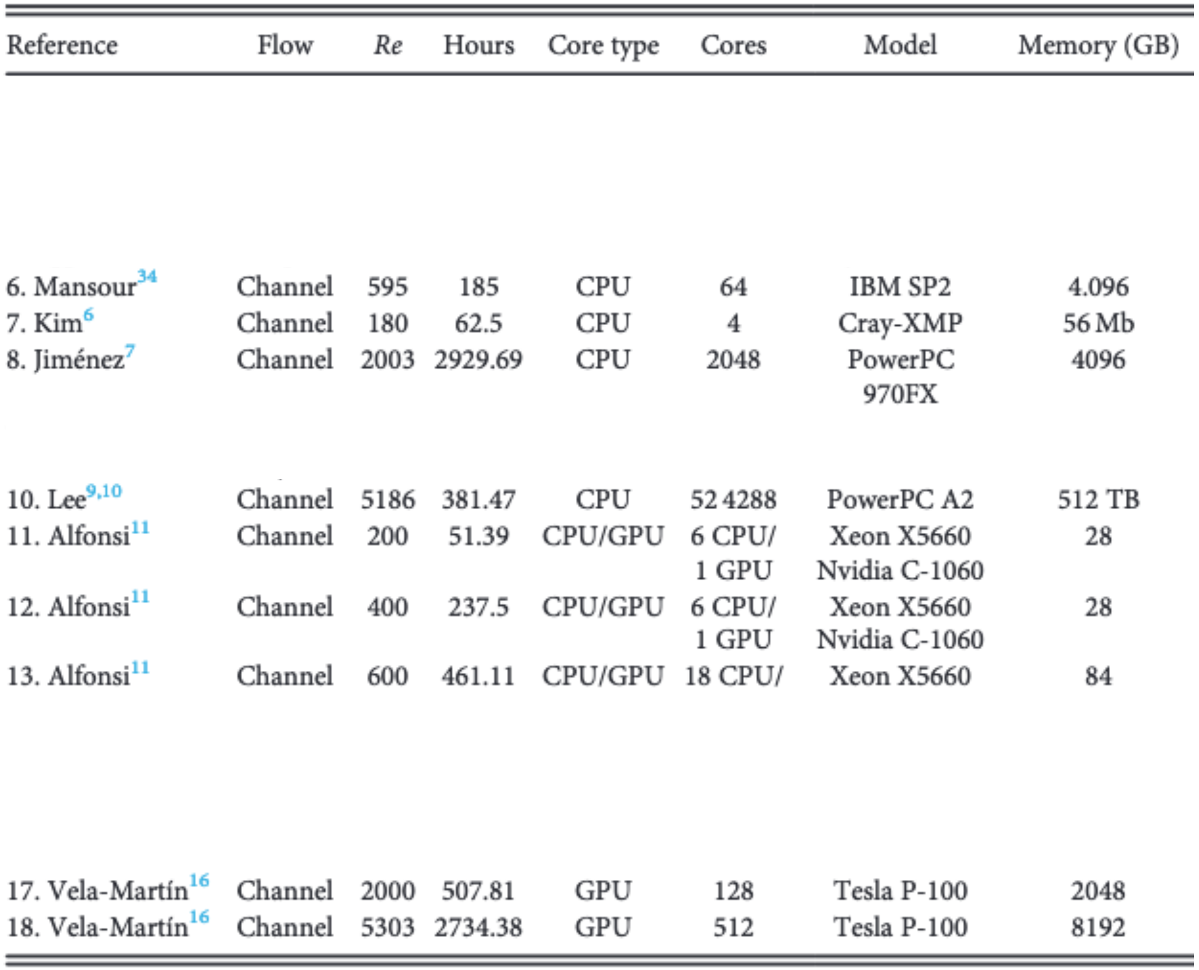
Target Final State to Achieve

But why do we care about this?

Computational Cost ~ $N_x \times N_y \times N_z$

Cost is estimated using: $N_{CPU} \times T$ (hours)



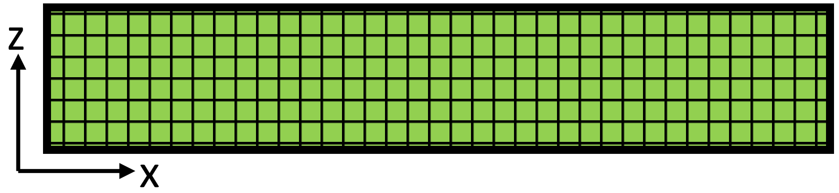


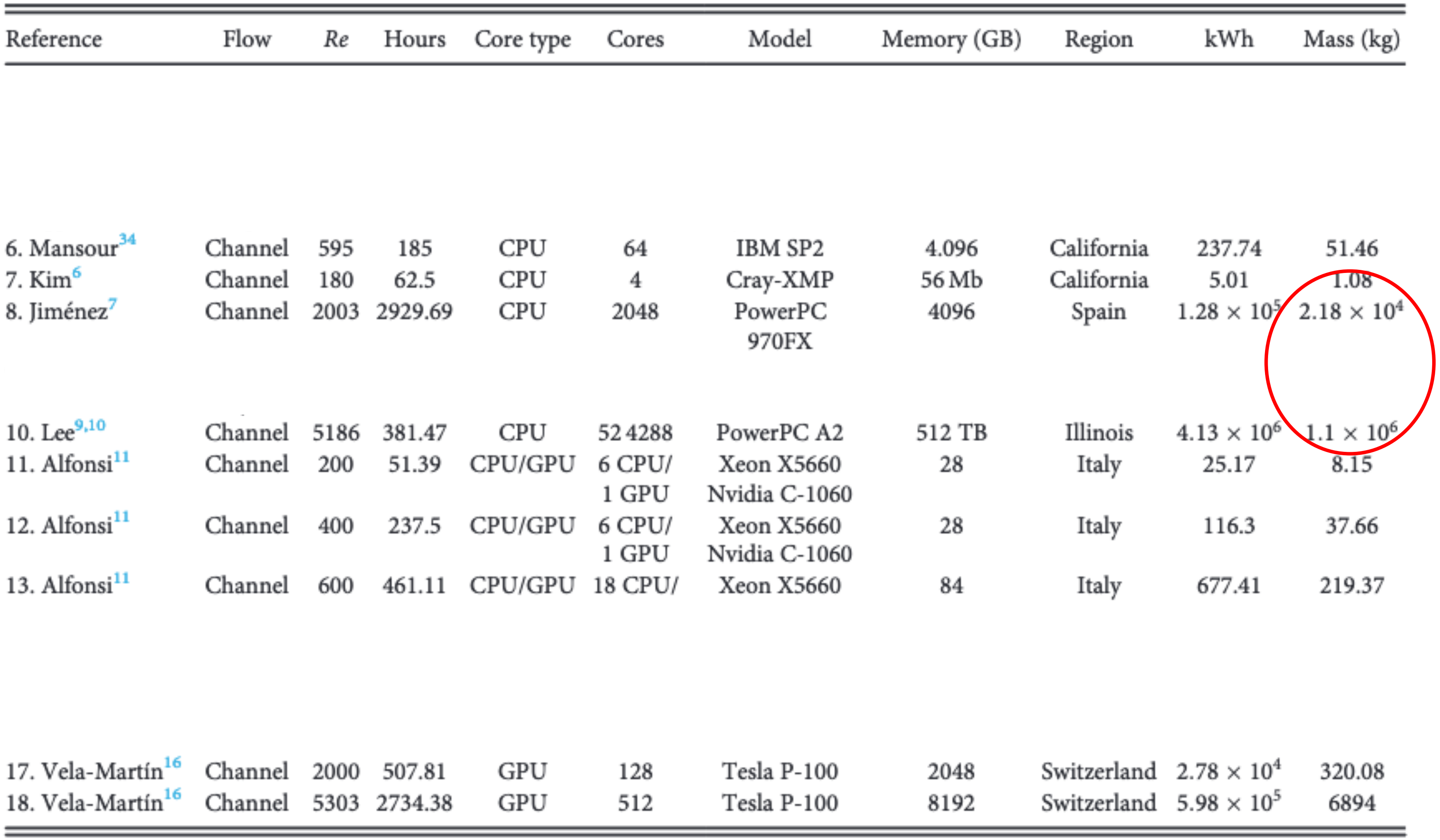
Reference	Flow	*Re*	Hours	Core type	Cores	Model	Memory (GB)
6. Mansour[34]	Channel	595	185	CPU	64	IBM SP2	4.096
7. Kim[6]	Channel	180	62.5	CPU	4	Cray-XMP	56 Mb
8. Jiménez[7]	Channel	2003	2929.69	CPU	2048	PowerPC 970FX	4096
10. Lee[9,10]	Channel	5186	381.47	CPU	52 4288	PowerPC A2	512 TB
11. Alfonsi[11]	Channel	200	51.39	CPU/GPU	6 CPU/ 1 GPU	Xeon X5660 Nvidia C-1060	28
12. Alfonsi[11]	Channel	400	237.5	CPU/GPU	6 CPU/ 1 GPU	Xeon X5660 Nvidia C-1060	28
13. Alfonsi[11]	Channel	600	461.11	CPU/GPU	18 CPU/	Xeon X5660	84
17. Vela-Martín[16]	Channel	2000	507.81	GPU	128	Tesla P-100	2048
18. Vela-Martín[16]	Channel	5303	2734.38	GPU	512	Tesla P-100	8192

But why do we care about this?

Computational Cost ~ $N_x \times N_y \times N_z$

Cost is estimated using: $N_{CPU} \times T$ (hours)



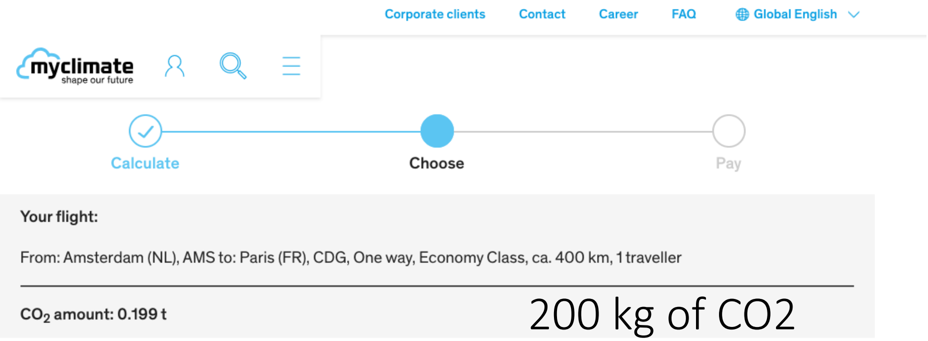


Reference	Flow	*Re*	Hours	Core type	Cores	Model	Memory (GB)	Region	kWh	Mass (kg)
6. Mansour[34]	Channel	595	185	CPU	64	IBM SP2	4.096	California	237.74	51.46
7. Kim[6]	Channel	180	62.5	CPU	4	Cray-XMP	56 Mb	California	5.01	1.08
8. Jiménez[7]	Channel	2003	2929.69	CPU	2048	PowerPC 970FX	4096	Spain	1.28×10^5	2.18×10^4
10. Lee[9,10]	Channel	5186	381.47	CPU	52 4288	PowerPC A2	512 TB	Illinois	4.13×10^6	1.1×10^6
11. Alfonsi[11]	Channel	200	51.39	CPU/GPU	6 CPU/ 1 GPU	Xeon X5660 Nvidia C-1060	28	Italy	25.17	8.15
12. Alfonsi[11]	Channel	400	237.5	CPU/GPU	6 CPU/ 1 GPU	Xeon X5660 Nvidia C-1060	28	Italy	116.3	37.66
13. Alfonsi[11]	Channel	600	461.11	CPU/GPU	18 CPU/	Xeon X5660	84	Italy	677.41	219.37
17. Vela-Martín[16]	Channel	2000	507.81	GPU	128	Tesla P-100	2048	Switzerland	2.78×10^4	320.08
18. Vela-Martín[16]	Channel	5303	2734.38	GPU	512	Tesla P-100	8192	Switzerland	5.98×10^5	6894

But why do we care about this?

Computational Cost ~ $N_x \times N_y \times N_z$

Cost is estimated using: $N_{CPU} \times T$ (hours)



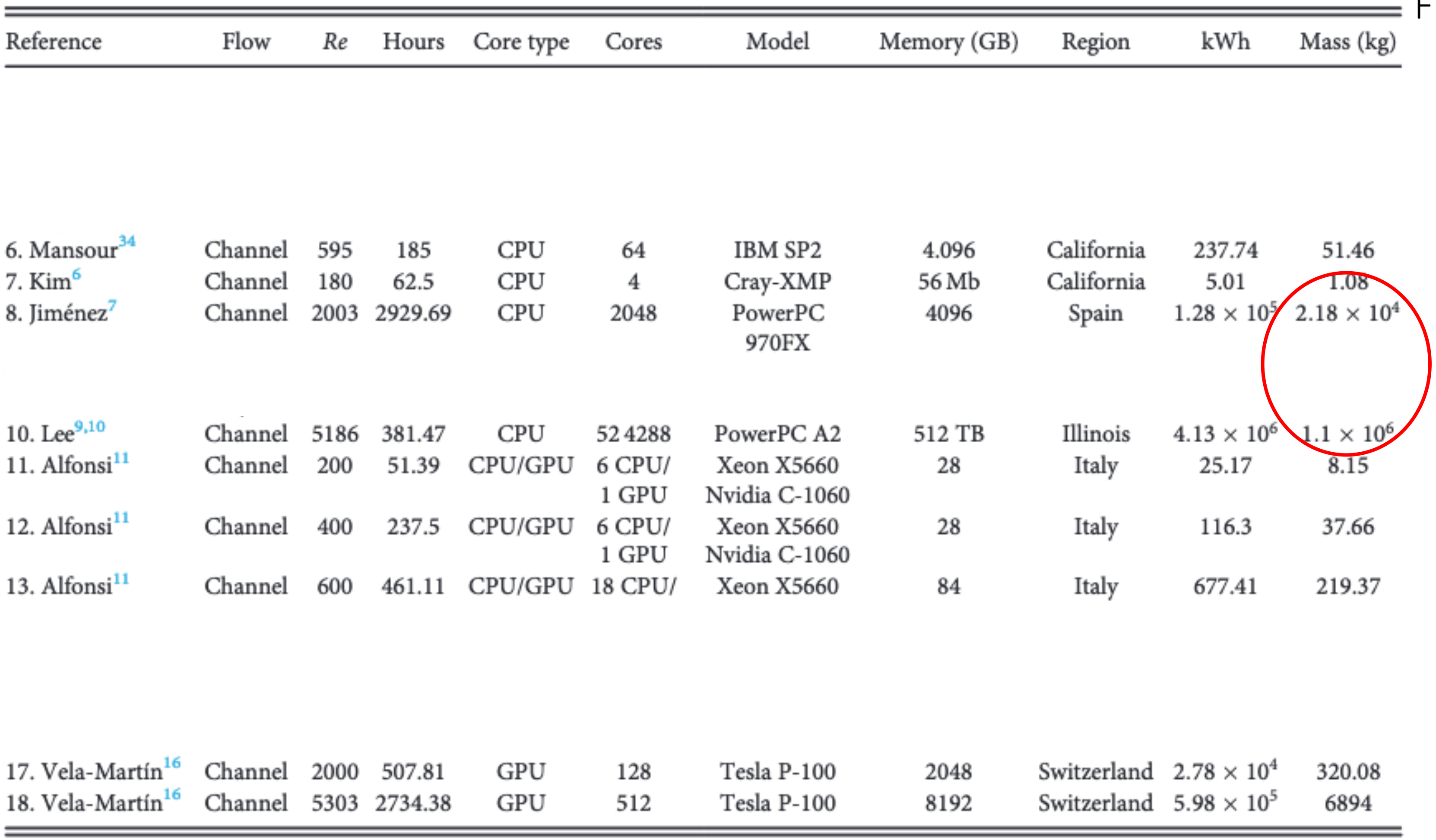
Your flight:

From: Amsterdam (NL), AMS to: Paris (FR), CDG, One way, Economy Class, ca. 400 km, 1 traveller

CO$_2$ amount: 0.199 t

200 kg of CO2

Flight from Amsterdam - Paris

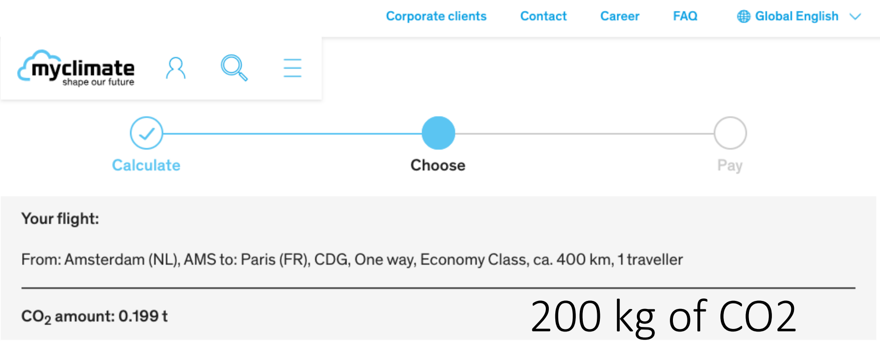


Reference	Flow	Re	Hours	Core type	Cores	Model	Memory (GB)	Region	kWh	Mass (kg)
6. Mansour[34]	Channel	595	185	CPU	64	IBM SP2	4.096	California	237.74	51.46
7. Kim[6]	Channel	180	62.5	CPU	4	Cray-XMP	56 Mb	California	5.01	1.08
8. Jiménez[7]	Channel	2003	2929.69	CPU	2048	PowerPC 970FX	4096	Spain	1.28×10^5	2.18×10^4
10. Lee[9,10]	Channel	5186	381.47	CPU	52 4288	PowerPC A2	512 TB	Illinois	4.13×10^6	1.1×10^6
11. Alfonsi[11]	Channel	200	51.39	CPU/GPU	6 CPU/ 1 GPU	Xeon X5660 Nvidia C-1060	28	Italy	25.17	8.15
12. Alfonsi[11]	Channel	400	237.5	CPU/GPU	6 CPU/ 1 GPU	Xeon X5660 Nvidia C-1060	28	Italy	116.3	37.66
13. Alfonsi[11]	Channel	600	461.11	CPU/GPU	18 CPU/	Xeon X5660	84	Italy	677.41	219.37
17. Vela-Martín[16]	Channel	2000	507.81	GPU	128	Tesla P-100	2048	Switzerland	2.78×10^4	320.08
18. Vela-Martín[16]	Channel	5303	2734.38	GPU	512	Tesla P-100	8192	Switzerland	5.98×10^5	6894

But why do we care about this?

Computational Cost ~ $N_x \times N_y \times N_z$

Cost is estimated using: $N_{CPU} \times T$ (hours)



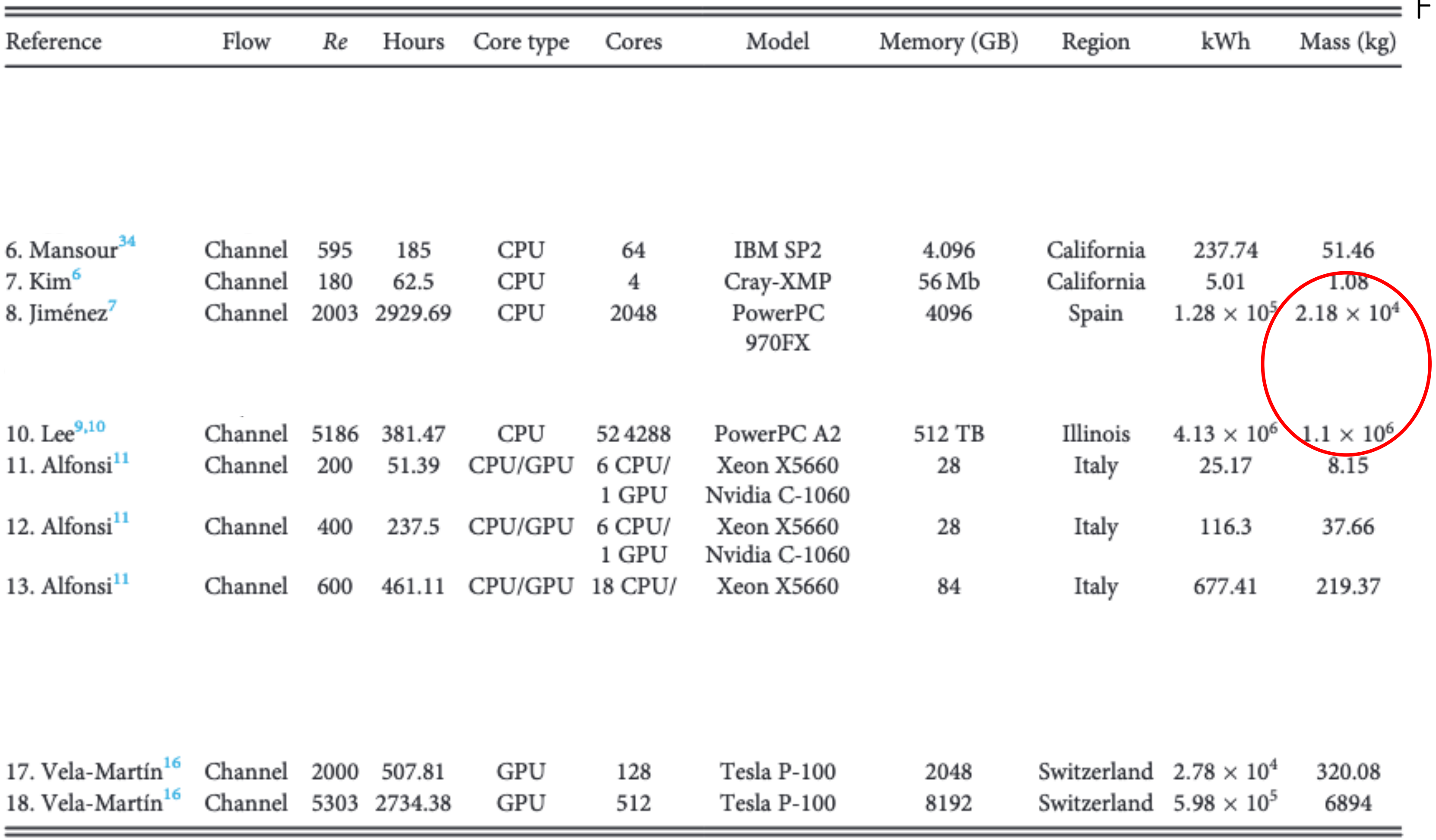
Your flight:

From: Amsterdam (NL), AMS to: Paris (FR), CDG, One way, Economy Class, ca. 400 km, 1 traveller

CO_2 amount: 0.199 t

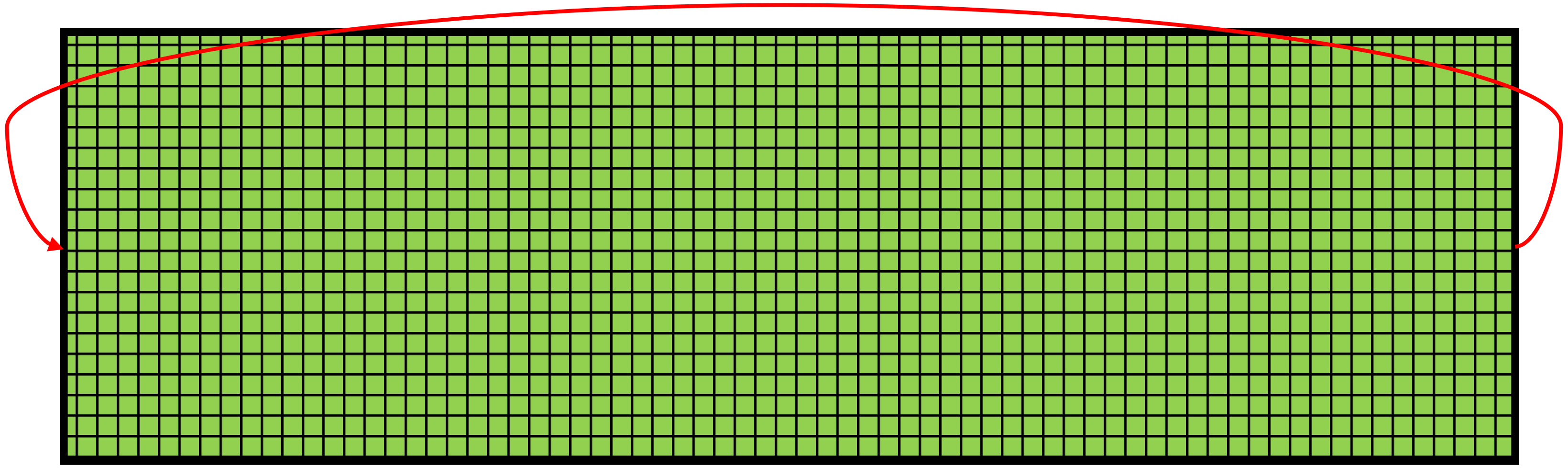
200 kg of CO2

Flight from Amsterdam - Paris

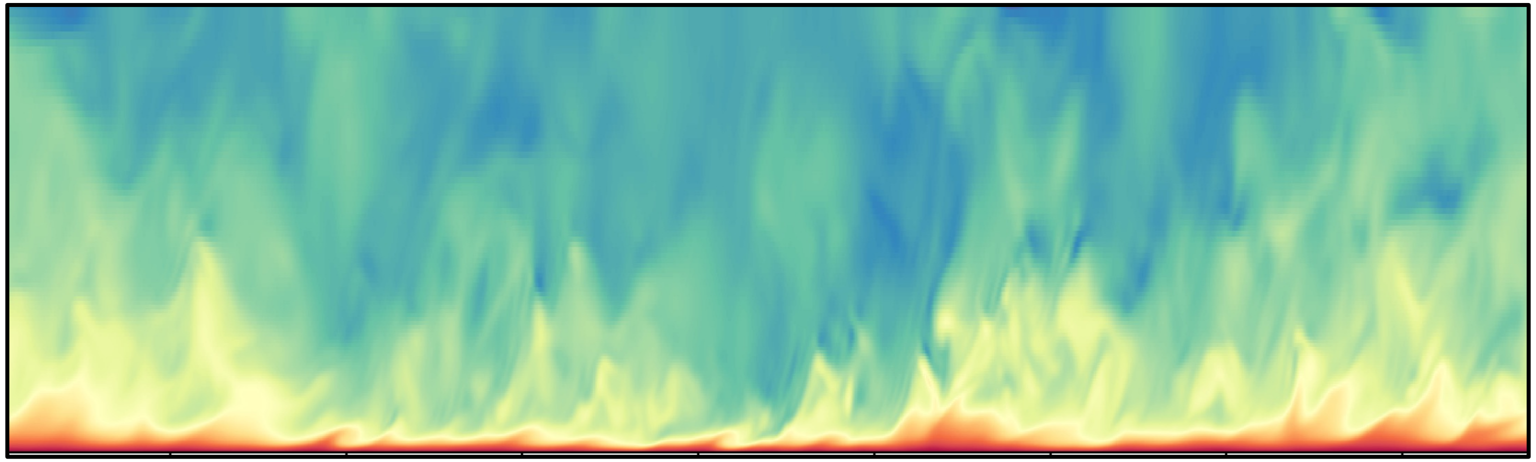


Reference	Flow	Re	Hours	Core type	Cores	Model	Memory (GB)	Region	kWh	Mass (kg)
6. Mansour[34]	Channel	595	185	CPU	64	IBM SP2	4.096	California	237.74	51.46
7. Kim[6]	Channel	180	62.5	CPU	4	Cray-XMP	56 Mb	California	5.01	1.08
8. Jiménez[7]	Channel	2003	2929.69	CPU	2048	PowerPC 970FX	4096	Spain	1.28×10^5	2.18×10^4
10. Lee[9,10]	Channel	5186	381.47	CPU	52 4288	PowerPC A2	512 TB	Illinois	4.13×10^6	1.1×10^6
11. Alfonsi[11]	Channel	200	51.39	CPU/GPU	6 CPU/ 1 GPU	Xeon X5660 Nvidia C-1060	28	Italy	25.17	8.15
12. Alfonsi[11]	Channel	400	237.5	CPU/GPU	6 CPU/ 1 GPU	Xeon X5660 Nvidia C-1060	28	Italy	116.3	37.66
13. Alfonsi[11]	Channel	600	461.11	CPU/GPU	18 CPU/	Xeon X5660	84	Italy	677.41	219.37
17. Vela-Martín[16]	Channel	2000	507.81	GPU	128	Tesla P-100	2048	Switzerland	2.78×10^4	320.08
18. Vela-Martín[16]	Channel	5303	2734.38	GPU	512	Tesla P-100	8192	Switzerland	5.98×10^5	6894

Lots of CO2 output for no reason! We better find a fast way to spin up to the final state.



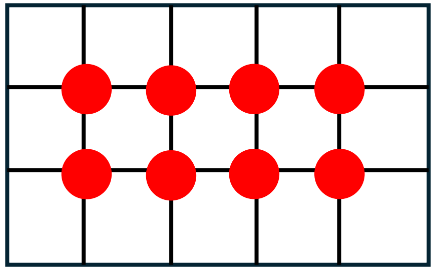
Standard Flow Configuration

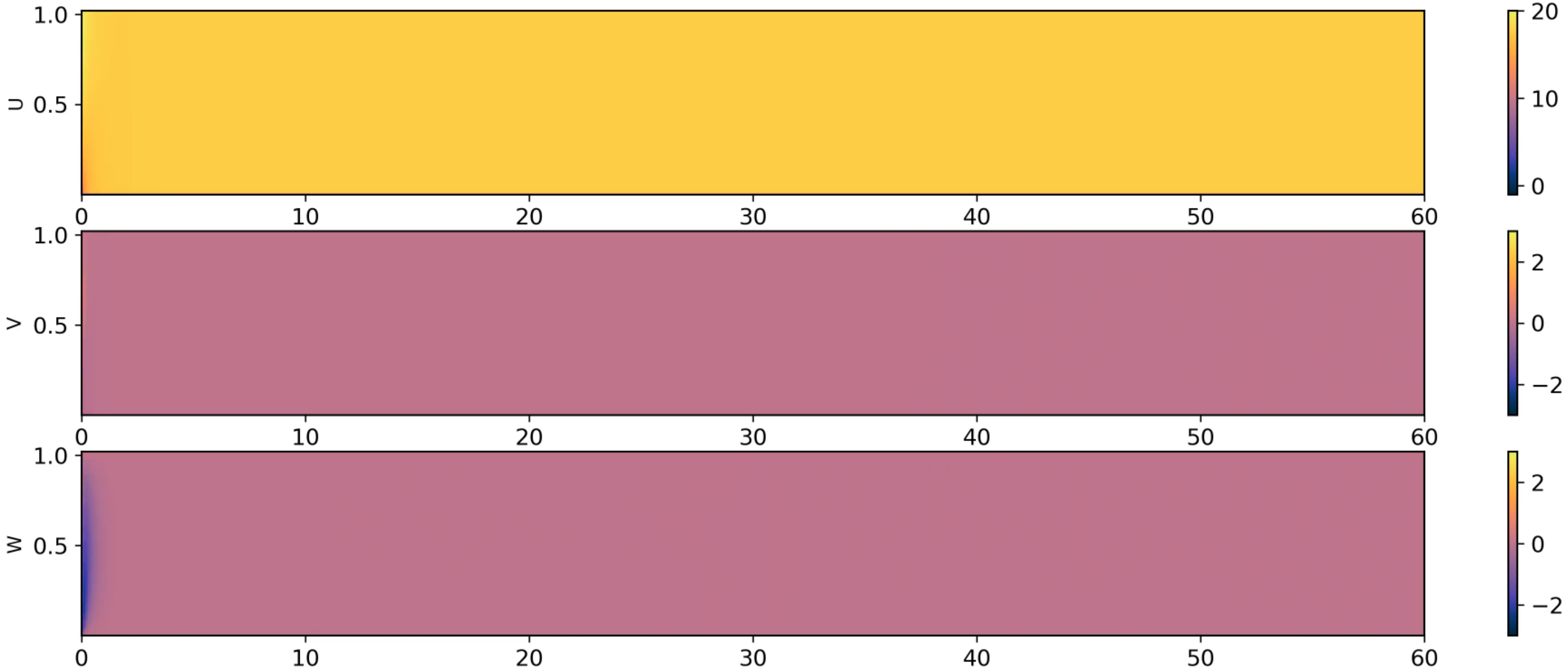


Target Final State to Achieve

Use an existing tool – differently! – No Math Version

Instead of using the synthetic turbulence generator as an inflow, populate the grid with data!

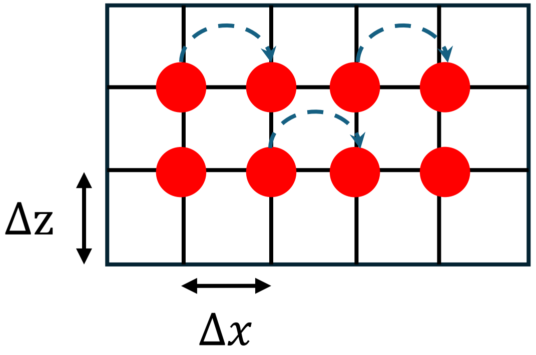




Time: 16.9 seconds.

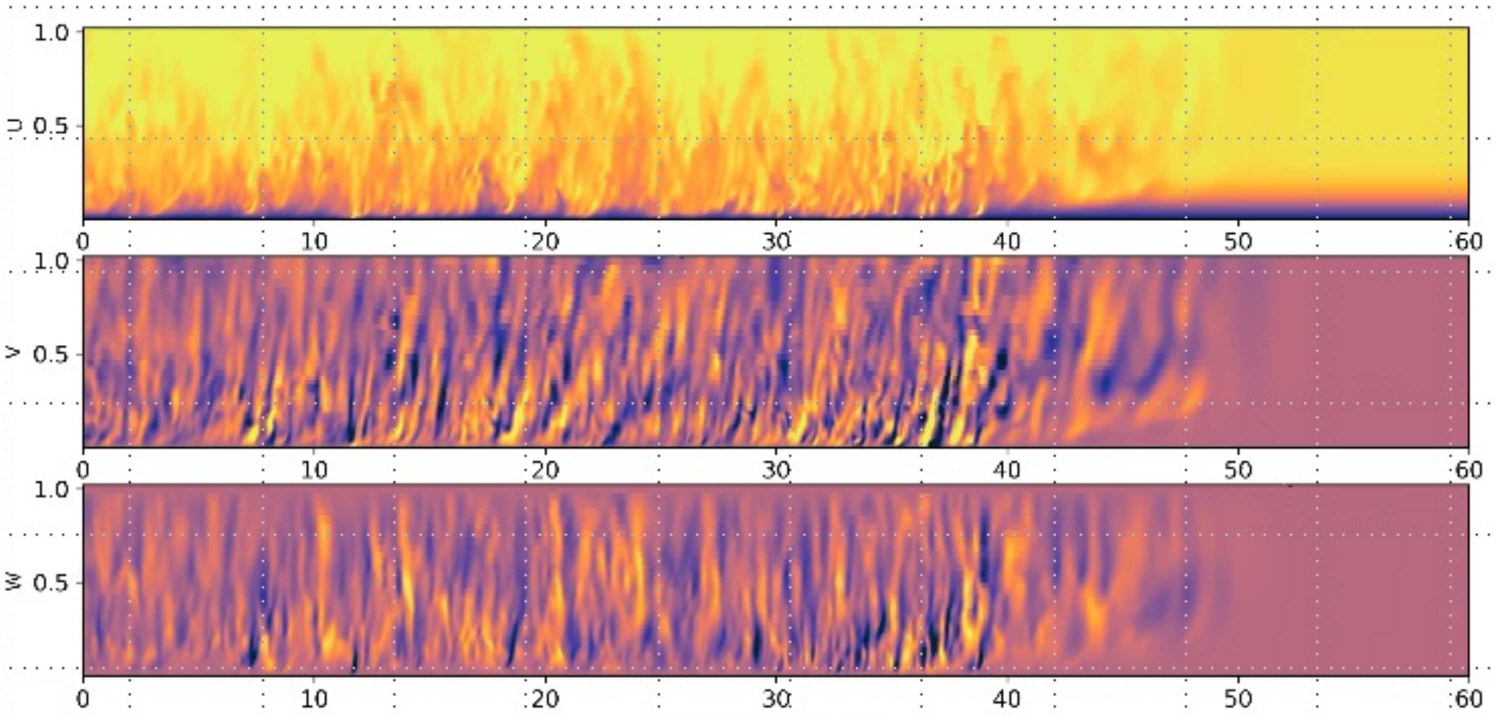
Use an existing tool – differently! – No Math Version

Instead of using the synthetic turbulence generator as an inflow, populate the grid with data!



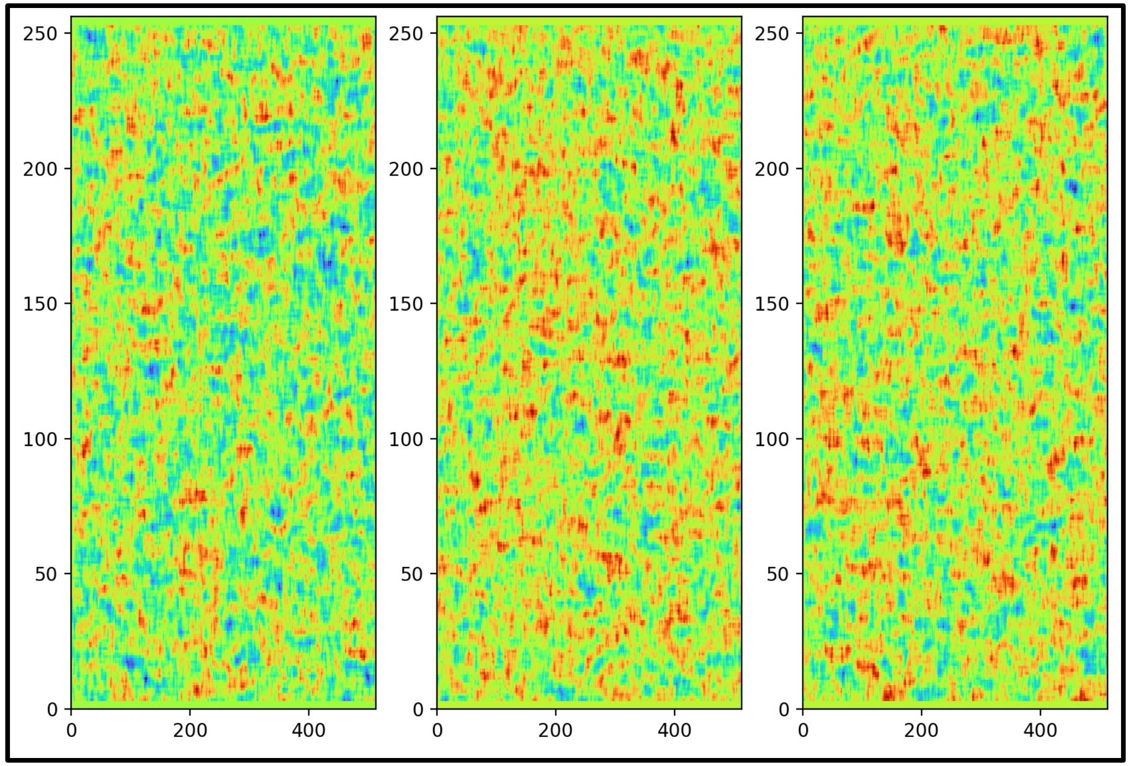
$$\xrightarrow{\text{Choose a sensible velocity}} \quad \Delta t_{\mathrm{STFG}} = \frac{\mathrm{CFL}\,\Delta x_1}{\boxed{U_c}},$$

Vertically averaged bulk velocity

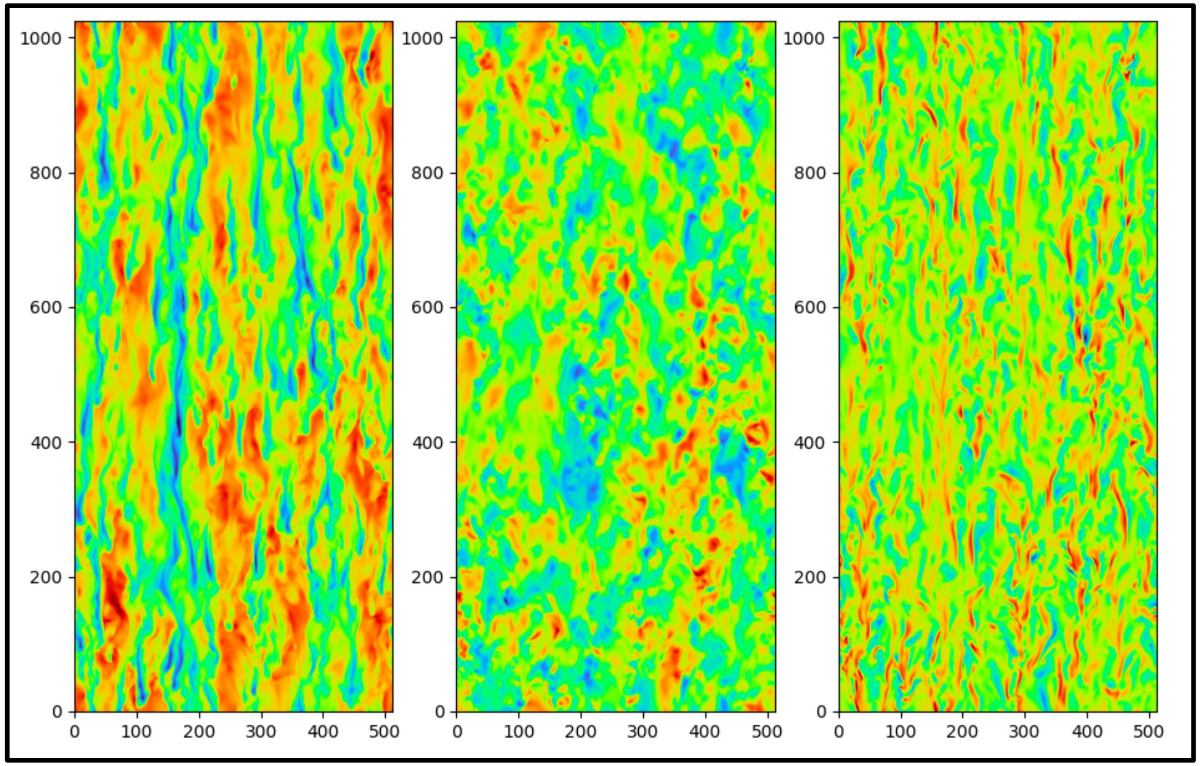


Can you guess which one is fake and which one is real?

(a)

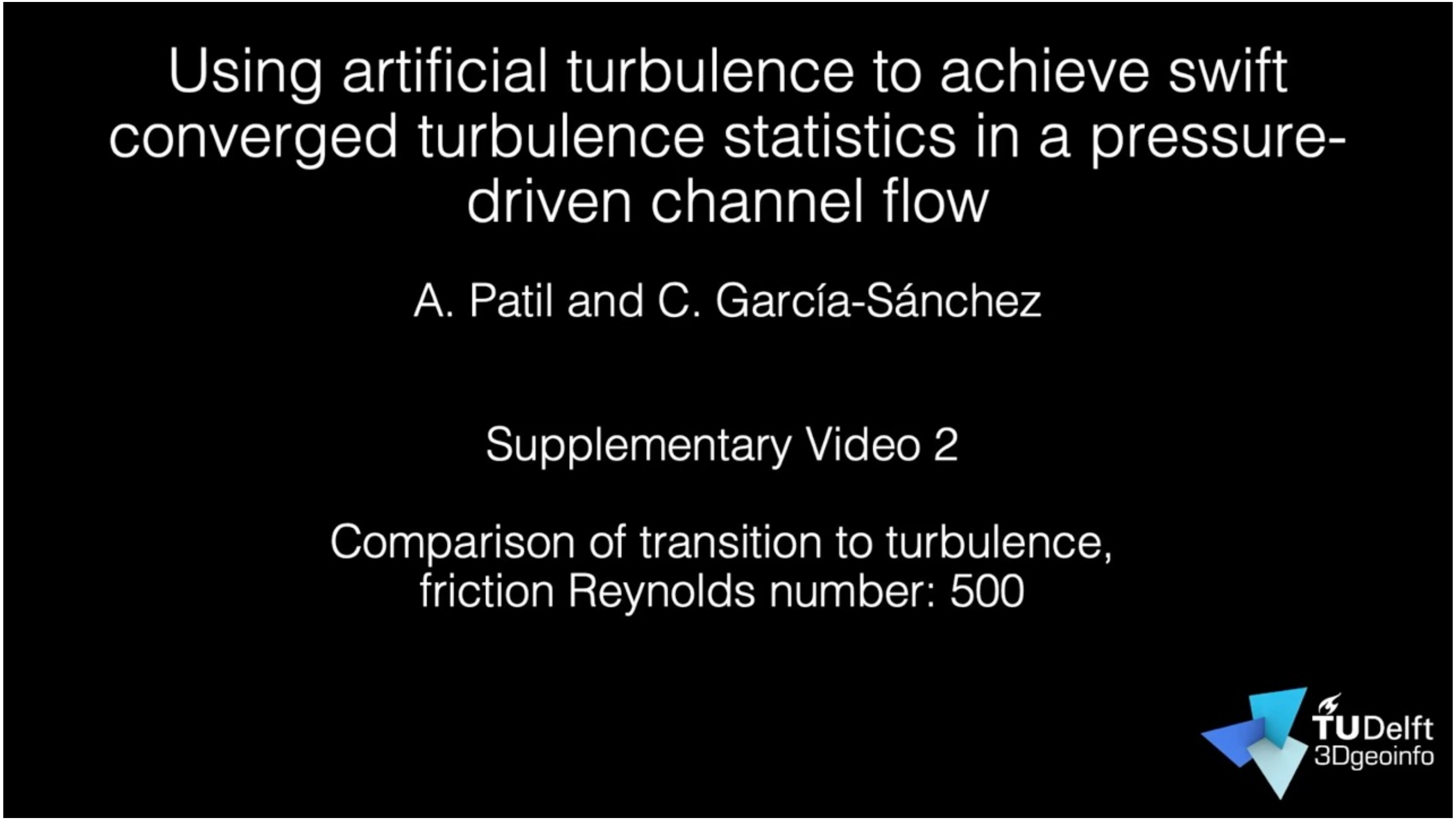


(b)



BreatheLab affiliates are NOT allowed to answer/respond!

Let’s test how good is this fake turbulence

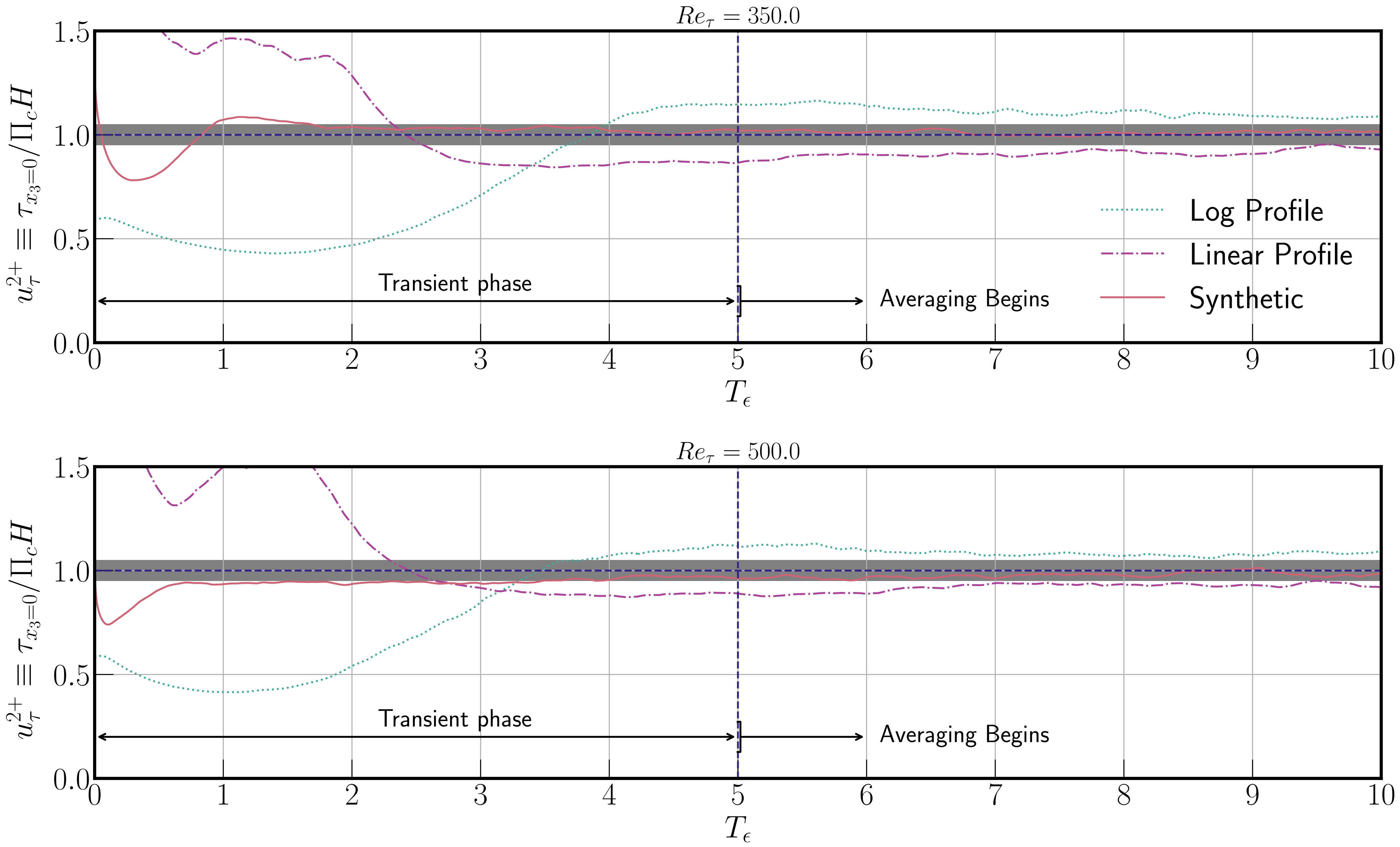


Using artificial turbulence to achieve swift converged turbulence statistics in a pressure-driven channel flow

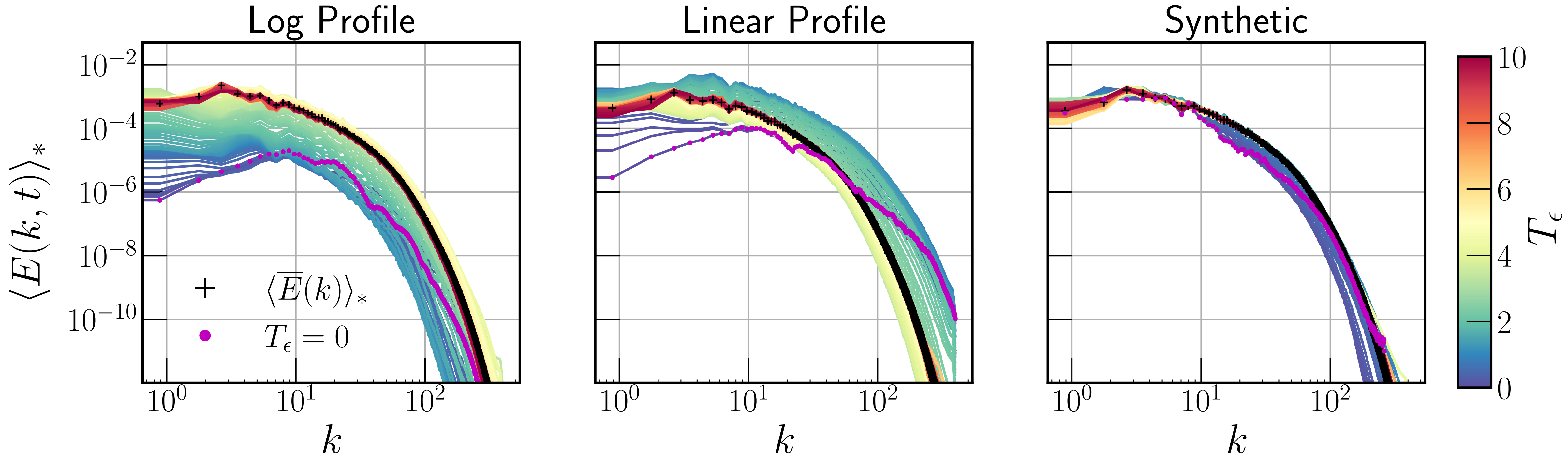
A. Patil and C. García-Sánchez

Supplementary Video 2

Comparison of transition to turbulence, friction Reynolds number: 500



$Re_\tau = 350.0$
$u_\tau^{2+} \equiv \tau_{x_3=0}/\Pi_c H$
1.5
1.0
0.5
0.0
0
1
2
3
4
5
6
7
8
9
10
T_ϵ
Transient phase
Averaging Begins
Log Profile
Linear Profile
Synthetic
$Re_\tau = 500.0$
$u_\tau^{2+} \equiv \tau_{x_3=0}/\Pi_c H$
1.5
1.0
0.5
0.0
0
1
2
3
4
5
6
7
8
9
10
T_ϵ
Transient phase
Averaging Begins



A factor of 7-8 faster compared to conventional methods

End of Part 1

Part 2: GenSDF

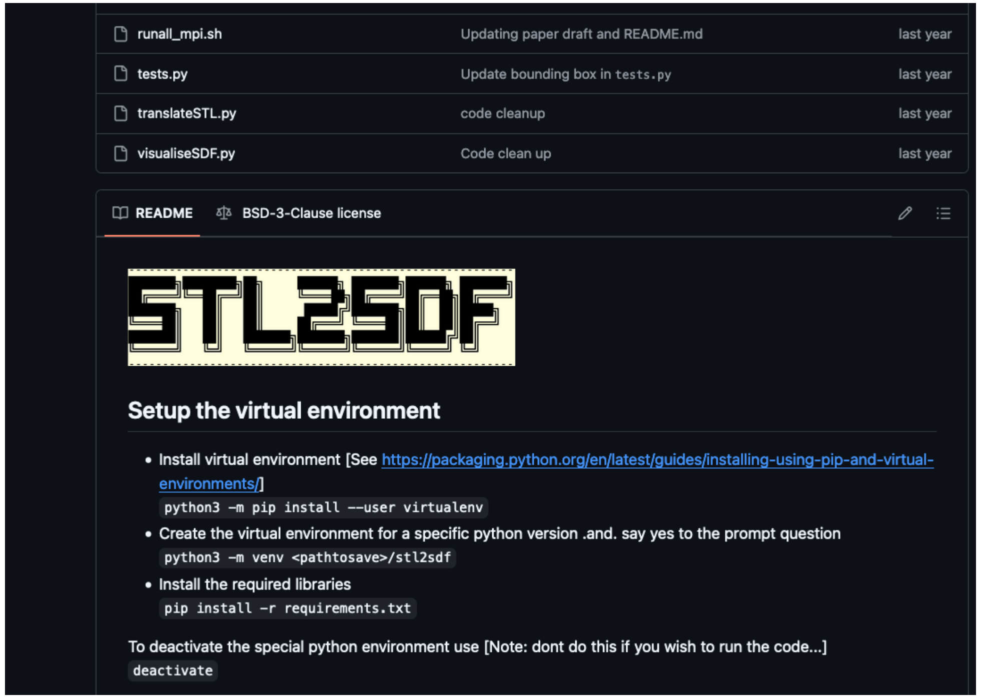
But Why?

Memory hungry trimesh library
Slow python code and many more……

Solution

Modern Fortran + MPI based implementation

- Low memory requirements
- Easy to port to existing CFD solvers in Fortran
- Good excuse to program :)



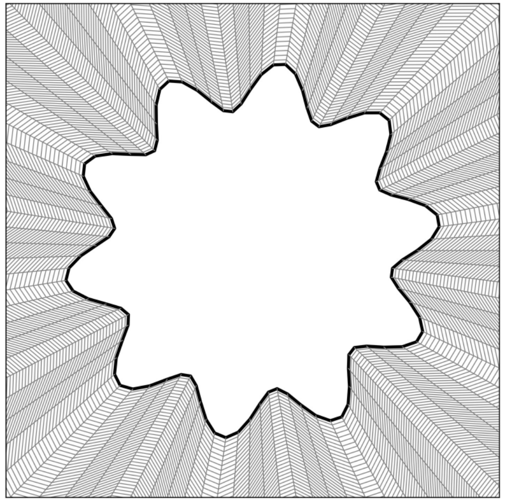
runall_mpi.sh	Updating paper draft and README.md	last year
tests.py	Update bounding box in `tests.py`	last year
translateSTL.py	code cleanup	last year
visualiseSDF.py	Code clean up	last year

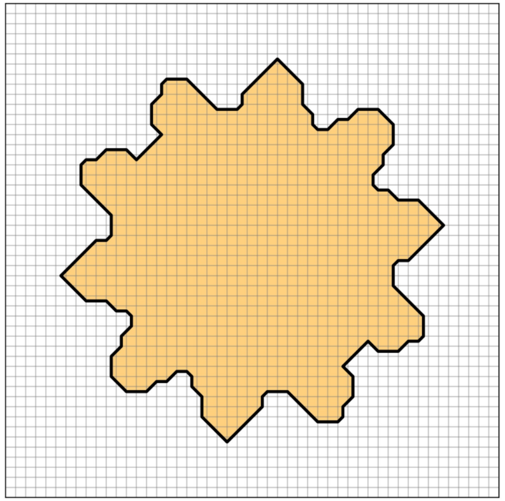
STL2SDF

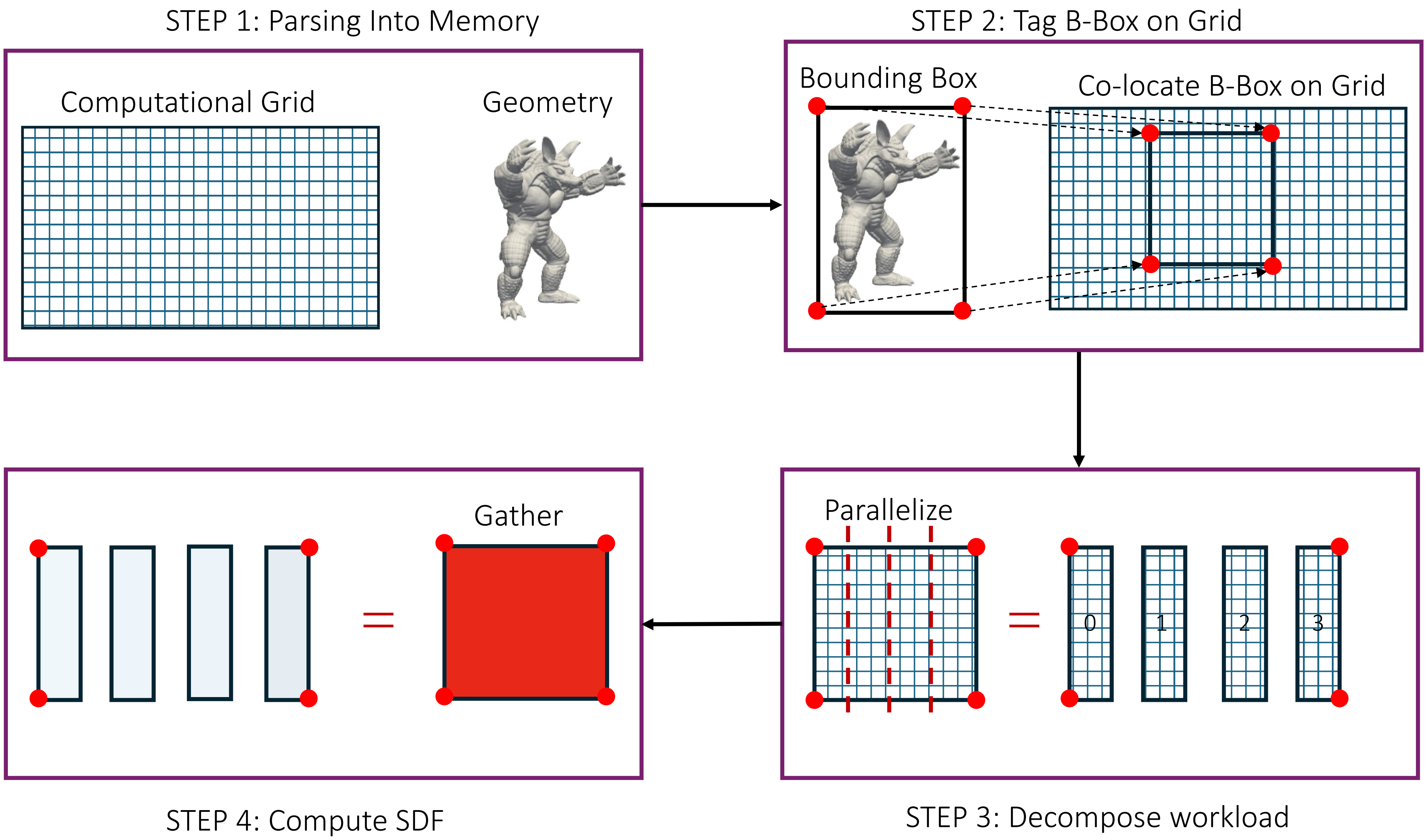
Setup the virtual environment

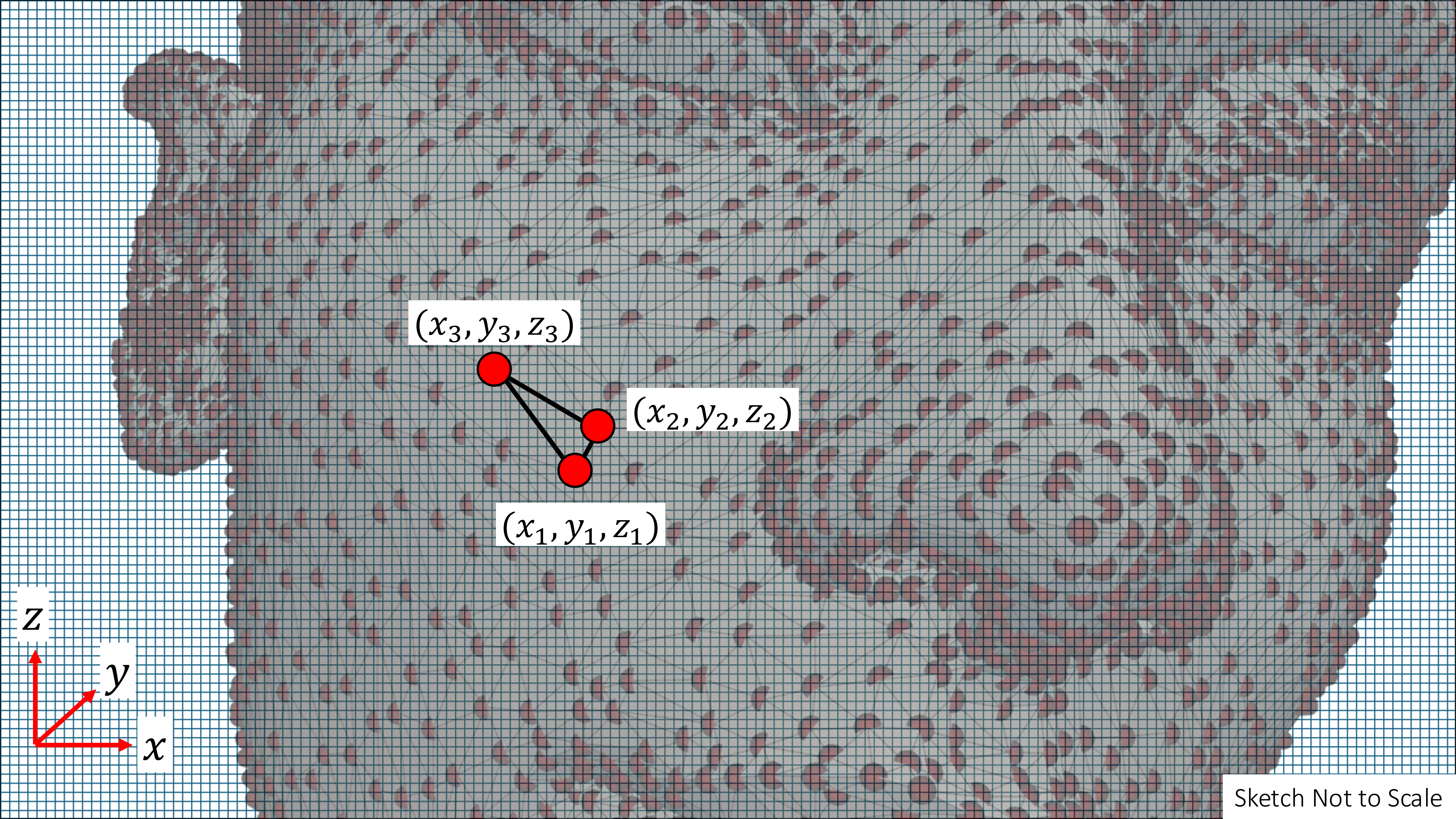
- Install virtual environment [See https://packaging.python.org/en/latest/guides/installing-using-pip-and-virtual-environments/]
  `python3 -m pip install --user virtualenv`
- Create the virtual environment for a specific python version .and. say yes to the prompt question
  `python3 -m venv <pathtosave>/stl2sdf`
- Install the required libraries
  `pip install -r requirements.txt`

To deactivate the special python environment use [Note: dont do this if you wish to run the code...]
`deactivate`



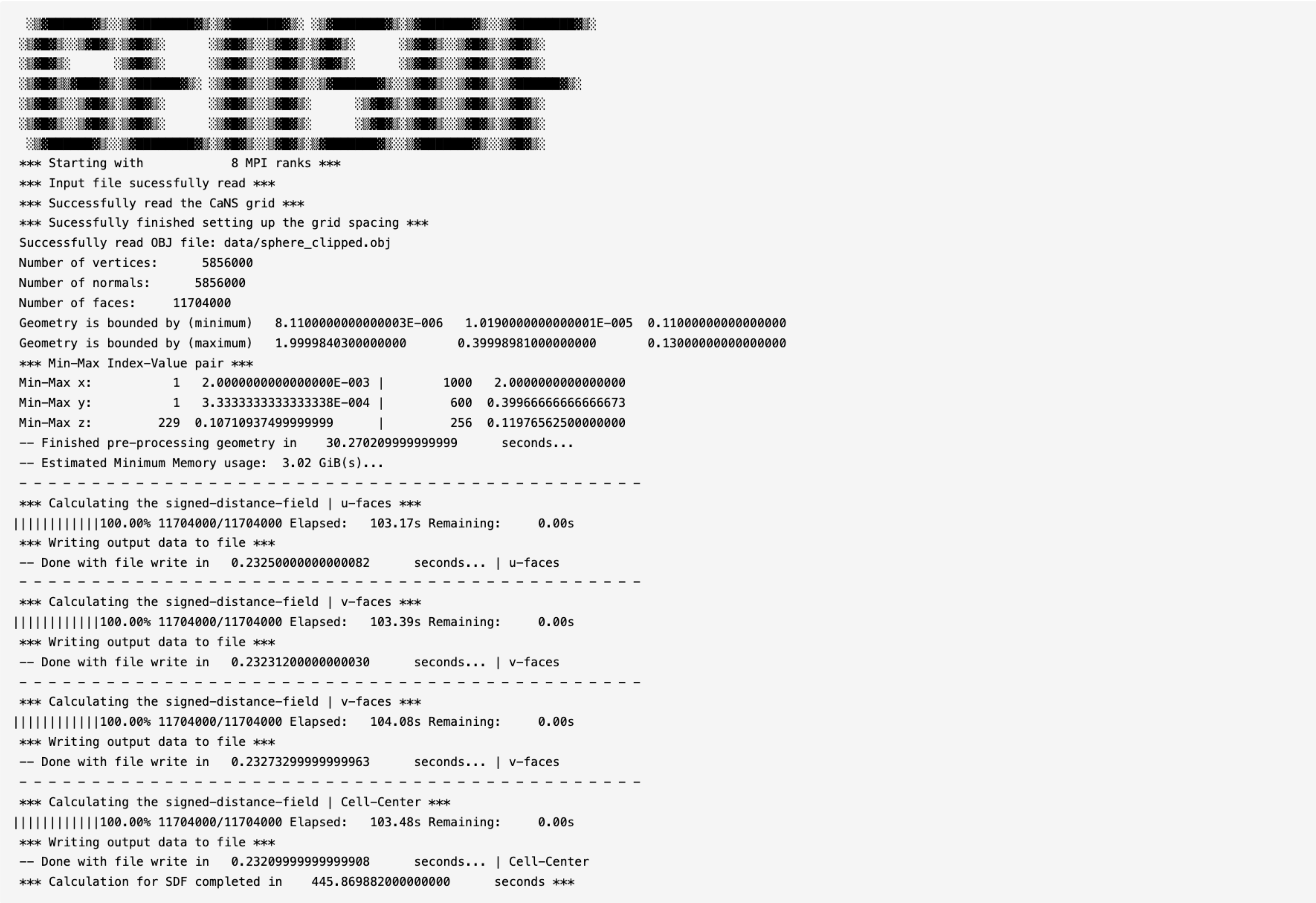



STEP 1: Parsing Into Memory
Computational Grid
Geometry
STEP 2: Tag B-Box on Grid
Bounding Box
Co-locate B-Box on Grid
Parallelize
=
0
1
2
3
STEP 3: Decompose workload
Gather
=
STEP 4: Compute SDF

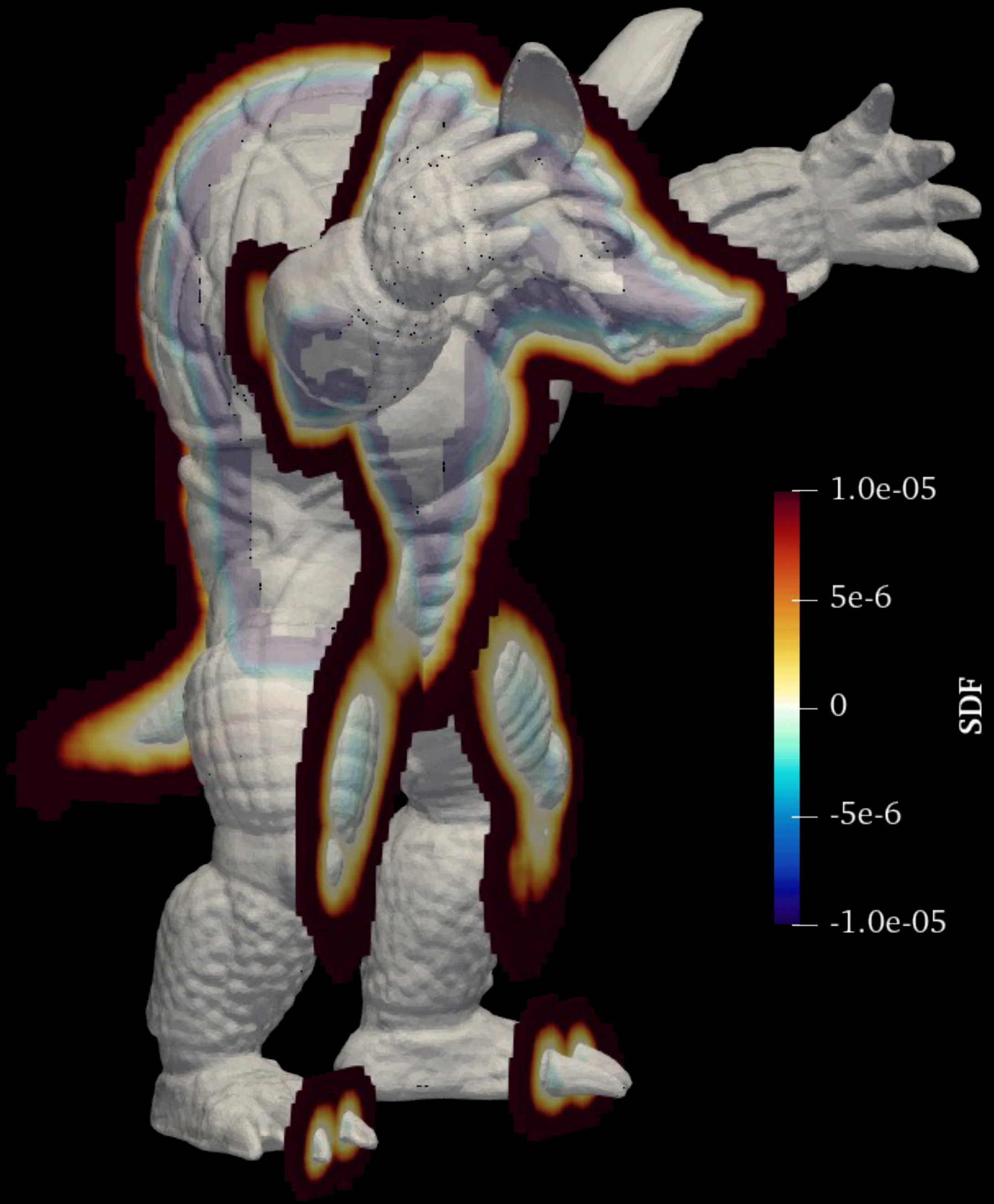


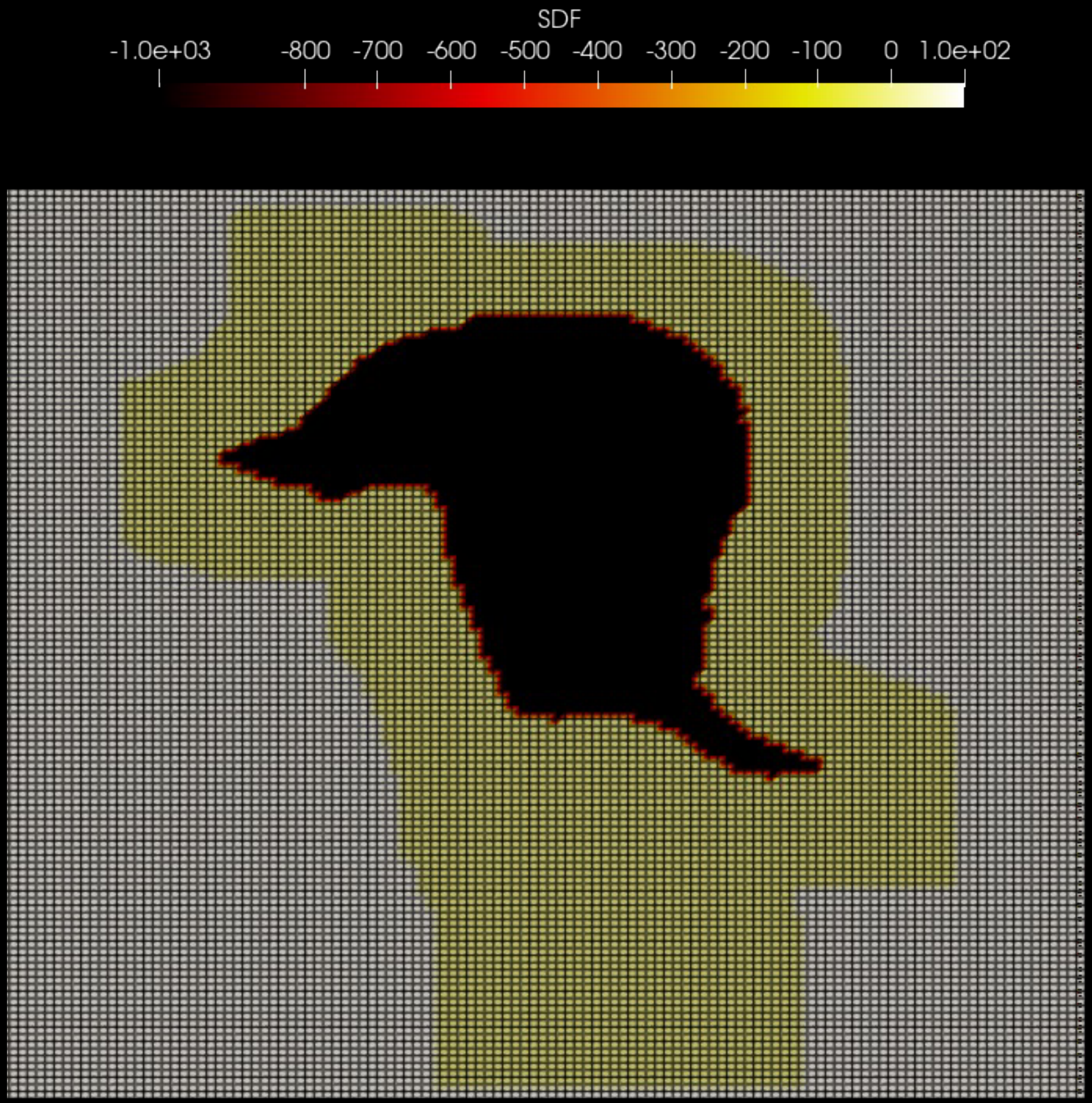
Sketch Not to Scale

18:04 | testrun | > mpirun -np 8 ./gensdf_mpi



```
 ░▒▓██████▓▒░▒▓████████▓▒░▒▓███████▓▒░ ░▒▓███████▓▒░▒▓███████▓▒░▒▓████████▓▒░
░▒▓█▓▒░▒▓█▓▒░▒▓█▓▒░     ░▒▓█▓▒░▒▓█▓▒░▒▓█▓▒░     ░▒▓█▓▒░▒▓█▓▒░▒▓█▓▒░
░▒▓█▓▒░     ░▒▓█▓▒░     ░▒▓█▓▒░▒▓█▓▒░▒▓█▓▒░     ░▒▓█▓▒░▒▓█▓▒░▒▓█▓▒░
░▒▓█▓▒▒▓███▓▒░▒▓██████▓▒░ ░▒▓█▓▒░▒▓█▓▒░░▒▓██████▓▒░▒▓█▓▒░▒▓█▓▒░▒▓██████▓▒░
░▒▓█▓▒░▒▓█▓▒░▒▓█▓▒░     ░▒▓█▓▒░▒▓█▓▒░     ░▒▓█▓▒░▒▓█▓▒░▒▓█▓▒░▒▓█▓▒░
░▒▓█▓▒░▒▓█▓▒░▒▓█▓▒░     ░▒▓█▓▒░▒▓█▓▒░     ░▒▓█▓▒░▒▓█▓▒░▒▓█▓▒░▒▓█▓▒░
 ░▒▓██████▓▒░▒▓████████▓▒░▒▓█▓▒░▒▓█▓▒░▒▓███████▓▒░▒▓███████▓▒░▒▓█▓▒░
 *** Starting with           8 MPI ranks ***
 *** Input file sucessfully read ***
 *** Successfully read the CaNS grid ***
 *** Sucessfully finished setting up the grid spacing ***
 Successfully read OBJ file: data/sphere_clipped.obj
 Number of vertices:      5856000
 Number of normals:      5856000
 Number of faces:     11704000
 Geometry is bounded by (minimum)   8.1100000000000003E-006   1.0190000000000001E-005  0.11000000000000000
 Geometry is bounded by (maximum)   1.9999840300000000       0.39998981000000000       0.13000000000000000
 *** Min-Max Index-Value pair ***
 Min-Max x:           1   2.0000000000000000E-003 |        1000   2.0000000000000000
 Min-Max y:           1   3.3333333333333338E-004 |         600  0.39966666666666673
 Min-Max z:         229  0.10710937499999999      |         256  0.11976562500000000
 -- Finished pre-processing geometry in    30.270209999999999      seconds...
 -- Estimated Minimum Memory usage:  3.02 GiB(s)...
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 *** Calculating the signed-distance-field | u-faces ***
|||||||||||100.00% 11704000/11704000 Elapsed:   103.17s Remaining:     0.00s
 *** Writing output data to file ***
 -- Done with file write in   0.23250000000000082      seconds... | u-faces
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 *** Calculating the signed-distance-field | v-faces ***
|||||||||||100.00% 11704000/11704000 Elapsed:   103.39s Remaining:     0.00s
 *** Writing output data to file ***
 -- Done with file write in   0.23231200000000030      seconds... | v-faces
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 *** Calculating the signed-distance-field | v-faces ***
|||||||||||100.00% 11704000/11704000 Elapsed:   104.08s Remaining:     0.00s
 *** Writing output data to file ***
 -- Done with file write in   0.23273299999999963      seconds... | v-faces
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 *** Calculating the signed-distance-field | Cell-Center ***
|||||||||||100.00% 11704000/11704000 Elapsed:   103.48s Remaining:     0.00s
 *** Writing output data to file ***
 -- Done with file write in   0.23209999999999908      seconds... | Cell-Center
 *** Calculation for SDF completed in    445.869882000000000      seconds ***
```


1.0e-05
5e-6
0
-5e-6
-1.0e-05
SDF


SDF
-1.0e+03
-800
-700
-600
-500
-400
-300
-200
-100
0
1.0e+02

Thank you!